基于深度强化学习的随机资源受限多项目动态调度策略

郭晓剑，胡方勇

(江西理工大学 经济管理学院，江西 赣州 341000)

摘　要：目前对于随机工期的分布式资源受限多项目调度(SDRCMPSP)问题的研究较少且大多数为静态调度方案，无法针对环境的变化实时地对策略进行调整优化，及时响应频繁发生的动态因素。为此建立了最小化总拖期成本为目标的随机资源受限多项目动态调度 DRL 模型，设计了相应的智能体交互环境，采用强化学习中的 DDDQN 算法对模型进行求解。实验首先对算法的超参数进行灵敏度分析，其次将最优组合在活动工期可变和到达时间不确定两种不同条件下对模型进行训练及测试，结果表明深度强化学习算法能够得到优于任意单一规则的调度结果，有效减少随机资源受限多项目期望总拖期成本，多项目调度决策优化提供良好的依据。

关键词：分布式多项目；随机调度；深度强化学习；资源约束

中图分类号：TP18　　doi: 10.19734/j.issn.1001-3695.2022.03.0065

Stochastic resource-constrained multi-project dynamic scheduling strategy based on deep reinforcement learning

Guo Xiaojian, Hu Fangyong

(*School of Economics & Management*, *Jiangxi University of Science & Technology*, *Ganzhou Jiangxi* 341000, *China*)

Abstract: There are few studies on the problem of stochastic resource-constrained distributed multi-project scheduling (SDRCMPSP) and most of them are static scheduling schemes, which cannot adjust and optimize the strategy in real time according to changes in the environment and respond to frequent dynamic factors in a timely manner. Therefore, this paper established a stochastic resource-constrained multi-project dynamic scheduling DRL model with the goal of minimizing the total drag cost, design the corresponding agent interaction environment, and use the DDDQN algorithm in reinforcement learning to solve the model. The experiment first analyzes the hyperparameters of the algorithm, and then trains and tests the model under two different conditions of variable activity duration and uncertain arrival time, and the results show that the deep reinforcement learning algorithm can obtain scheduling results that are better than any single rule, effectively reduce the total drag-off cost of random resources limited multi-project expectations, and provide a good basis for multi-project scheduling decision optimization.

Key words: distributed multi-project; stochastic scheduling; deep reinforcement learning; resource-constrained

0　引言

随着社会的发展项目管理技术显得尤为重要，在项目调度的过程中资源的合理配置与调度起着决定性的作用，该类问题通常称之为资源受限项目调度问题(resource-constrained project scheduling problem, RCPSP)[1]。现实情况下往往需要同时调度多个资源受限的项目[2]，且存在局部与全局资源的约束称之为分布式资源首先多项目调度(distributed resource-constrained multi-project scheduling problem,DRCMPSP)。然而在工程实际中，项目实施可能受到各种不确定因素的影响，如缺少相关项目经验、生产设备故障、资源不可用、天气状况等导致活动工期或项目到达时间与预估时间产生偏离[3]，使预先制定的多项目计划不可行，便需要一种有效的方法减少多项目的期望总拖期成本 ETTC(except total tardiness cost)该类问题称之为随机资源受限多项目调度(SDRCMPSP)问题。目前关于 SDRCMPSP 问题的文献相对较少，并且大部分调度策略是静态的[4]，如优先规则算法[5][6]。此外 Song 等采用优先规则启发式算法来生成基线计划，并将受影响的活动推至最早可行的时间执行[7]、Tosselli 等采用了重复协商博弈的方法[8]、刘东宁等采用了多优先规则启发式方法(MPRH)[9]，该方法虽然能够在活动工期变化的动态环境下减少多项目工期延误成本，但在每次决策点时需要进行多次仿真实验，未对以往经验进行有效的利用，无法及时对动态环境作出实时的决策。

以目标为导向的强化学习算法，以其能够实现离线学习与在线应用的优势被广泛运用于动态环境下的调度问题上[10]。且由以往文献可知强化学习算法在被广泛运用于车间作业的动态调度问题上并取得较好的结果[11][12][13][14]，且目前并未有文献研究深度强化学习算法在多项目随机调度问题上的应用，因此本文将该算法运用至 SDRCMPSP 的动态调度上。

本文针对活动工期、项目到达日期偏差情况而造成多项目作业的总拖期成本 TTC(total tardiness cost)增加的影响，通过深度强化学习中智能体不断与环境进行仿真交互，采用文献[14]的 DDDQN 算法进行调度策略的优化。首先提出静态环境下多项目调度的数学模型并将其结合并行调度方案转换为动态调度过程，其次根据多项目动态调度流程及总拖期成本搭建与智能体交互的环境，运用 DDDQN 算法使的智能体在环境中根据当前状态不断进行探索和对现有知识的利用优化不同状态下的策略，以此降低随机资源约束多项目调度的 ETTC。将所建立的模型与算法进行超参数的策略组合分析求出最优的超参数组合，后将该组合运用活动工期可变或项目到达时间不确定的动态环境下进行仿真研究。仿真结果表明应用深度强化学习方法能够使智能体学习到优于任何单一规则的调度策略，为多项目在动态环境下的调度提供良好的决策依据。

收稿日期：2022-03-01；**修回日期**：2022-04-15

作者简介：郭晓剑(1971-)，男，江西赣州人，副教授，硕士，主要研究方向为项目管理、BIM 技术人工智能；胡方勇(1999-)，男，浙江温州人，硕士研究生，主要研究方向为项目调度与管理(1443204684@qq.com).

1 问题描述

1.1 多项目静态问题

假设一个多项目是由 m 单一项目 i(i=1,..,m)所组成，在项目 i 中存在 a_{ij}(j=1,...,J_i)个实活动其工期为 d_{ij} 和两个工期及资源用量为零的虚拟活动 a_{i0}、$a_i(J_i+1)$。A_{ij} 表示活动 a_{ij} 的紧前活动集合，活动 a_{ij} 的开始时间需大于 A_{ij} 中活动完成时间的最大值；L_i 表示项目 i 的局部资源集合，R_{il} 表示局部资源的可用量；G 表示全局资源集合用 R_g 表示其可用量；活动 a_{ij} 在任意时刻被执行时，需要 r^l_{ij} 及 r^g_{ij} 的局部和全局可更新资源；项目 i 的到达时间为 ST_i，其完工日期以项目 i 的 $a_i(J_i+1)$活动的完工时间 $FT_i(J_i+1)$表示。$FT_i(J_i+1)$当超出截止日期时便会产生拖期成本 TC(tardiness cost)，对于项目 i 其延期成本 TC_i 为

$$TC_i = (FT_i(J_i+1) - ST_i - cpl_i) \times c_i \tag{1}$$

其中:c_i 表示项目 i 单位延期成本，$ST_i + cpl_i$ 表示项目 i 的截至日期。而在考虑全局资源分配的多项目调度问题中，其目标为最小化各项目的总拖期成本 TTC，即

$$\text{Min} \sum_{i=1}^{m} TC_i \tag{2}$$

1.2 多项目动态调度转换

然而在实际情况中活动的工期会受到环境的不确定性而产生变化服从一定的概率分布，此时多项目的静态调度问题转变为动态调度问题。由于串行调度尽早安排活动的原理，不适用于动态环境下多项目的调度，本文采用并行调度方式实现多项目的动态调度。在多项目的并行调度过程中 t 表示当前时间(多项目开工至今的时间且初始为 0)，存在的集合包括已完工活动集合 F、正在执行的活动的作业集合 D、所有紧前活动均已完工的候选活动集合 P、各项目未选择活动的集合 U，多项目动态调度流程如下：

a)输入多项目调度信息包括各项目的规模、活动工期、优先级关系、局部与全局资源需求量，清空所有活动集合。

b)判断所有未添加至 U 集合的项目的开始时间 ST_i 是否小于等于 t，若是则将对应项目的所有活动添加至 U。

c)判断 U 中是否存在所有紧前活动均已完工的活动，若存在则将其添加至P中并从U中删除该活动，否则转至步骤c)。

d)若 P 为非空集合：则根据调度规则从 P 中选取优先级最高的活动并判断所选活动与 D 中活动是否存在资源冲突，若是则转至步骤 d)否则将该活动从 P 中删除后添加至 D 中，并继续步骤 c)；若 P 为空集合：转至步骤 d)。

e)判断 D 中最早完工的一个或若干个活动，令 t 等于该些活动的完工时间将其从 D 中移除并添加至 F 中。

f)判断多项目的活动是否全部完成，若是则输出各项目的完工时间即 $FT_i(J_i+1)$(i=1,...,m)同时计算+TTC，否则转至步骤b)。

2 基于 DDDQN 的分布式多项目动态调度

多项目动态调度的问题是要从当前时刻各项目的可执行活动集中按照一定规则并在满足局部、全局资源约束的条件下选取优先执行的若干个活动进行作业，直到所有项目的活动均调度完成为止。这是一个连续的决策过程，因此可以将该问题转换为马尔可夫或半马尔可夫决策问题，即对问题的状态、动作、即时奖励进行定义。

2.1 状态描述

状态的特征定义应能够充分反映当前时刻智能体所处环境的局部和全局特征，当进行决策时，智能体需根据当前时刻的环境特征作出左右决策。当问题的状态集为有限个时，可以采取数组或表格的方式进行表示，被称之为 RL。然而在实际问题中往往具有较大或者连续的状态空间，此时需要采用深度神经网络函数逼近的能力，消除 RL 算法面临的“维度灾难问题”。在本文中多项目调度的状态特征包括三个 m×max(J_i+1)的矩阵，max(J_i+1)表示所有项目活动规模中的最大值，三个矩阵分别为调度结果矩阵 FN、活动作业矩阵 DN、可执行活动矩阵 PN。

FN 是由每个项目中各活动的完工时间组成的矩阵，在输入网络前对其进行最大值标准化处理，初始赋值为零。

$$FN = \begin{bmatrix} FT_{1,1} & \cdots & FT_{1,J_i-1} & \cdots & FT_{1,max(J_i-1)} \\ \vdots & & \vdots & & \vdots \\ FT_{i,1} & & FT_{i,J_i-1} & & FT_{1,max(J_i-1)} \\ \vdots & & \vdots & & \vdots \\ FT_{m,1} & \cdots & FT_{m,J_m-1} & \cdots & FT_{m,max(J_i-1)} \end{bmatrix} \tag{3}$$

DN 表示各项目中每个活动当前所处状态，若正在执行则为 1 否则为 0 由于该矩阵中的数值取值为 1 或 0 因此不需要规范化。

$$DN = \begin{bmatrix} 1 & \cdots & 1 & \cdots & 0 \\ \vdots & & \vdots & & \vdots \\ 1 & \cdots & 0 & \cdots & 0 \\ \vdots & & \vdots & & \vdots \\ 0 & \cdots & 0 & \cdots & 0 \end{bmatrix} \tag{4}$$

PN 表示各项目中每个活动是否在当前时刻执行(即该活动的所有紧前活动均已完成)若是则为 1 否则为 0，同理无须进行规范化。

$$PN = \begin{bmatrix} 0 & \cdots & 1 & \cdots & 0 \\ \vdots & & \vdots & & \vdots \\ 0 & \cdots & 1 & \cdots & 0 \\ \vdots & & \vdots & & \vdots \\ 0 & \cdots & 1 & \cdots & 0 \end{bmatrix} \tag{5}$$

考虑到深度学习原始输入中的特征提取，将三个特征矩阵看做高度为矩阵行数、宽度为矩阵列数的图像的三个不同通道，采用卷积输入的形式进行训练。

2.2 动作定义

在多项目动态调度 DRL 模型中，动作空间是由许多单一规则的调度算法组成，通过强化学习算法针对不同状态选择合适的调度规则，以此克服单一规则的局限性。本文选用 15 个单一的调度规则，前 7 个为集中式调度规则，后 8 个为复合式调度规则。其中 OFT_i 表示项目 i 在仅考虑局部资源约束下的最优调度方案，采用基于活动列表编码的改进灰狼算法求解所得。15 个调度规则如表 1 所示，其中项目 i 表示已经到达且候选活动集非空的项目，j 表示该项目中的候选活动。其中规则 1~6 将各项目的候选活动集视为一个集合在选择活动的同时确定了该活动所在项目，规则 7~14 则是优先先择项目 p 之后从项目 p 的候选活动集中选取活动。

2.3 奖励函数

由于本文研究的分布式多项目动态调度的目标时实现总拖期成本最小，为了即时奖励能够准确评价动作进行如下设定：

$$r_t = \frac{\sum_{i=1}^{m}\left(\sum_{j\in F_i^t} d_{ij} - \sum_{j\in F_i^{t-1}} d_{ij}\right)}{\sum_{i=1}^{m}\sum_{j=1}^{J_i+1} d_{ij}} + \frac{(\max\{FT_i^{t-1}\} - \max\{FT_i^t\})\times\omega_i}{\sum_{i=1}^{m}\sum_{j=1}^{J_i+1} d_{ij}} \tag{6}$$

其中:max$\{FT_i^t\}$表示 t 时刻项目 i 已完工活动的最大完工时间，ω_i 表示项目拖期成本。令

$$u_t = \frac{\sum_{i=1}^{m}\left(\sum_{j\in F^t} d_{ij} - \max\{FT_i^t\}\times\omega_i\right)}{\sum_{i=1}^{m}\sum_{j=1}^{J_i+1} d_{ij}} \tag{7}$$

则 u_0=0 此时算法的累计奖励计算如下：

$$R = \sum_t^T r_t = \sum_{t=1}^{T} u_{t-1} - u_t = u_0 - u_1 + u_1 - u_2 + \cdots + u_{T-1} - u_T =$$
$$u_0 - u_T = \frac{\sum_{i=1}^{m}\sum_{j=1}^{J_i} d_{ij} - \max\{FT_i^T\}\times\omega_i}{\sum_{i=1}^{m}\sum_{j=1}^{J_i+1} d_{ij}} \tag{8}$$

因此累计奖励 R 最大化同等与 u_T 最小，由于各活动的预计工期 d_{ij} 均为常数，则 u_T 最小同等与各项目完工工期与拖期成本乘积之和最小即 TTC 最小。

表 1　动作空间

Tab. 1　Action space

序号	名称	公式
1	SOF	$\text{job}=min\{d_{ij}\}$
2	LOF	$\text{job}=\max\{d_{ij}\}$
3	MINLFT	$\text{job}=\min\{\text{LF}T_{ij}\}$
4	MINOFT	$\text{job}=\min\{\text{OFT}_{ij}\}$
5	WMDD	$\text{job}=\min\{\max\{\text{LFT}_{ij}-t,d_{ij}\}/w_i\}$
6	WMDD2	$\text{job}=\min\{\max\{\text{OFT}_{ij}-t,d_{ij}\}/w_i\}$
7	MINSLK	$\text{job}=\min\{\text{LS}_{ij}-\max\{ES_{ij},t\}\}$
8	MINLT+OFT	$p=\min\{\text{F}T_i^t-LFT_i\},job=\min\{OFT_{pj}\}$
9	MINLT+LFT	$p=\min\{\text{F}T_i^t-LFT_i\},job=\min\{\text{L}FT_{pj}\}$
10	MAXLT+OFT	$p=\max\{\text{F}T_i^t-LFT_i\},job=\min\{OFT_{pj}\}$
11	MAXLT+LFT	$p=\max\{\text{F}T_i^t-LFT_i\},job=\min\{\text{L}FT_{pj}\}$
12	MINTC+OFT	$p=\min\{w_i\},job=\min\{\text{O}FT_{pj}\}$
13	MINTC+LFT	$p=\min\{w_i\},job=\min\{\text{L}FT_{pj}\}$
14	MAXTC+OFT	$p=\max\{w_i\},job=\min\{\text{O}FT_{pj}\}$
15	MAXTC+LFT	$p=\max\{w_i\},job=\min\{\text{L}FT_{pj}\}$

2.4　探索利用策略

合理的探索利用策略能够使得智能体充分利用所学到的经验知识，同时保证能够探索新的策略行为。本文采用的探索利用策略为线性递减的贪婪策略，智能体的动作策略如下：

$$a=\begin{cases}\arg\max_{a'}Q(a') & \text{rand}()<\varepsilon\\ random & \text{rand}()>\varepsilon\end{cases}\tag{9}$$

其中:rand()为一个[0,1]的随机数，ε 为贪婪策略的概率，服从以下分布：

$$\varepsilon^t=\min(\varepsilon_{\min},\varepsilon^{t-1}\times\varepsilon_{rate})\tag{10}$$

其中:ε_{min} 为 ε 的最小值，ε_{rate} 为衰减率。

2.5　DDDQN 算法流程

a)定义算法折扣因学习率 α，经验池容量 M，网络训练周期 L，目标网络更新周期 N，最小训练批量 mini_batch，最大训练回合 T_max，初始化 Q 网络与目标 Q 网络参数 θ、θ'，输入多项目调度信息，令 step =0。

b)重置各项目调度计划信息并清除调度结果集，初始化多项目调度状态 s_0。

c)根据当前状态 state 依据探索利用策略选择当前决策点智能体的 action 选择的优先规则调度算法，进行多项目调度流程中的步骤 b)~e)。

d)根据式(6)计算当前时刻的即时奖励值 reward 与下一时刻状态 state_以及训练是否结束标志 done(结束为 True,未结束为 False)，将{state,action,reward,state_,done}五元组储存至经验池中。

e)判断是否满足网络参数更新条件，若是则根据 TD 误差对网络参数进行更新，否则转至步骤 f)。

f)判断所有项目是否全部完工，若是则令 step+=1 转至步骤 g)，否则转至步骤 c)。

g)判断是否达到最大训练回合即 step=T_max，若是则停止训练并将 Q 网络参数保存本地，否则转至步骤 b)。

算法 1　基于 DDDQN 的多项目动态调度伪代码

初始化最小训练批量 mini_batch, step-size η，经验池容量 M，网络训练周期 L 和目标网络更新周期 N，最大训练周期 T_max，动作选择次数 num=0。

随机初始化 Q 网络参数 θ 并将其参数复制给目标 Q'网络 θ'

for step= 1 **to** M **do**

重置多项目调度信息、清除调度结果，初始化调度状态 s1。

While 多项目活动未全部完工 **then**

基于探索利用策略选择动作及对应的优先规则,num+=1。

根据最大优先规则执行多项目调度中的步骤 2、3、4 直到产生资源冲突。

计算当前奖励 rt 和下一状态 st+1，判断 done 是否为 True。

将(st, at,rt,st+1,done) 储存至经验池 M。

if num % L==0 and num>mini_batch **then**

从经验池中随机选取 mini_batch 个(st, at,rt,st+1,done)。

for j = 1 **to** mini_batch **do**

$$令\ y_j=\begin{cases}r_j & \text{if done}=\text{True}\\ r_j+\gamma Q'(s_{j+1},\arg\max_a Q(s_{j+1},a;\theta),\theta') & \text{otherwise}\end{cases}$$

end for

更根据 y 与 Q(s,a;θ)采用反向传播更新 Q 网络参数 θ

end if

if num % N==0 **then**

将 Q 网络的参数 θ 复制给目标 Q 网络 θ'←θ

end if

break

end for

3　仿真实验

为验证所搭建仿真环境的有效性以及 DDDQN 算法解决对分布式多项目动态调度问题的有效性，选取 MPSPLIB 标准库中的 MP30_5 和 MP90_2 问题集进行测试，各问题集均含有 5 个算例具体信息如表 2 所示。实验在配备 Windows 10 64 位系统、24GB 运行内存、处理器为 AMD R7 4800H 的笔记本上搭建 tensflow2.0 环境下运行。

表 2　问题集的具体信息

Tab. 2　Specific information of question set

信息	MP30_5	MP90_2
项目数	5	2
活动数	30	90
问题规模	150	180
平均资源利用系数	0.82	0.57

同时多项目中活动工期服从常见概率分布类型如表 3 所示。

表 3　常见工期分布

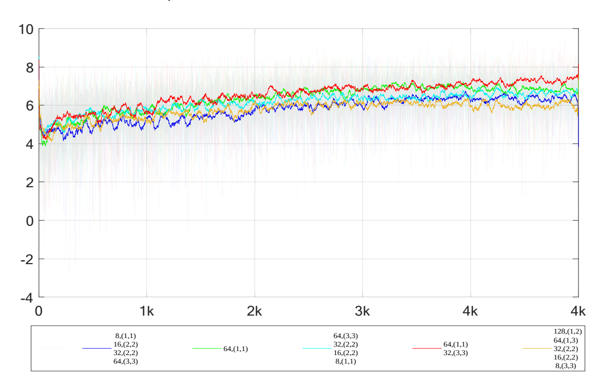
Tab. 3　Common duration distribution

名称	分布类型	区间	方差
U1	均匀分布	$\left[d_{ij}-\sqrt{d_{ij}},d_{ij}+\sqrt{d_{ij}}\right]$	$d_{ij}/3$
U2	均匀分布	$\left[0,2d_{ij}\right]$	$d_{ij}^2/3$
EXP	指数分布	$d_{ij}e^{-d_{ij}}$	d_{ij}^2
B1	β 分布	$\alpha=d_{ij}/2-1/3,\beta=2\alpha$	$d_{ij}/3$
B2	β 分布	$\alpha=1/6,\beta=2\alpha$	$d_{ij}^2/3$

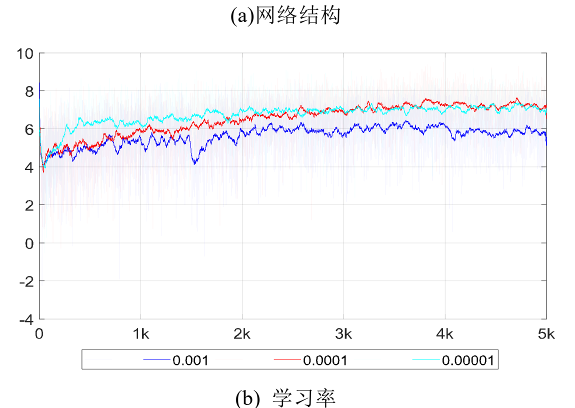
3.1　参数分析

在强化学习中超参数对于网络学习性能至关重要，目前对于超参数的确定一般依靠人工经验和随机搜索。为确定最优的超参数组合，本文选用算例 MP30_5_5 在工期为 U1 分布类型对模型的网络结构、速率 rate、目标网络更新周期 N、最小训练批量 mini_batch、折扣率 γ 等超参数进行了灵敏度分析，同时在对某一超参数进行灵敏度分析时其他超参数的取值均保持不变。图 1 为 DRL 模型的超参数在不同取值下的累计奖励迭代图，通过训练过程中累计奖励的变化来判断该超参数取值的效果。以图 1(a)为例，其横坐标为模型的训练次数，纵坐标为累计奖励的变化曲线。从该图中可以看出不同网络结构能够影响算法的性能，当网络结构取值为红线所示时算法的性能最佳，将其确定为本文模型的网络结构，此时其他参数均保持不变。同理可得模型的其他超参数的取值，最终确定超参数策略如表 4 所示。

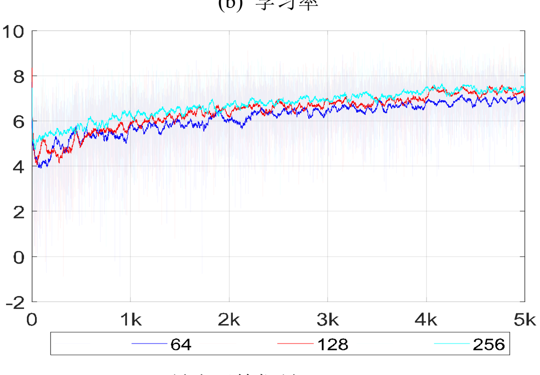
图 1(a)表示五种不同卷积层对算法的影响，每一行表示滤波器数量和内核大小其中步幅均为(1，1)。由图中可以看出第 4 种网络结构相对于其他四种，能够为算法带来更有效的性能提升。图 1(b)表示不同的学习率对于算法的影响，可以看当 rate 较低时算法训练性能最差，而 rate 较高时算法性能有所下降，因此本文选取学习率为 0.0001。图 1(c)表示不同的目标网络更新周期对算法的影响，从图中可以看出当 N=100 周期的训练性能最好。图 1(d)可以看出最小训练批量对算法的影响较小，当 mini_batch 为 64 时算法的性能会有所下降，当 mini_batch 为 256 或 128 时算法的收敛趋势较为稳定，但由于 256 需要更多的训练时间因此选用 128。图 1(e)表示不同的折扣系数对于算法的影响，同理当 γ 较低时会降低算法性能，当 γ 较高时算法收敛较慢，选取 0.99 折扣率。



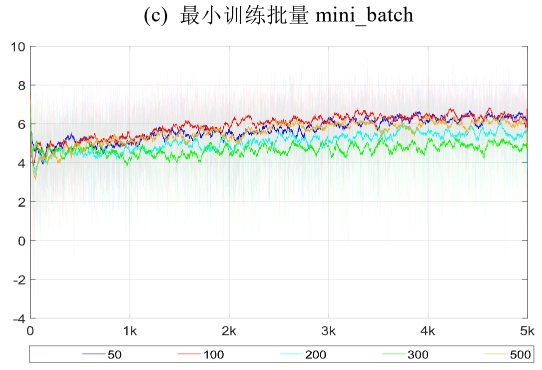
(a)网络结构



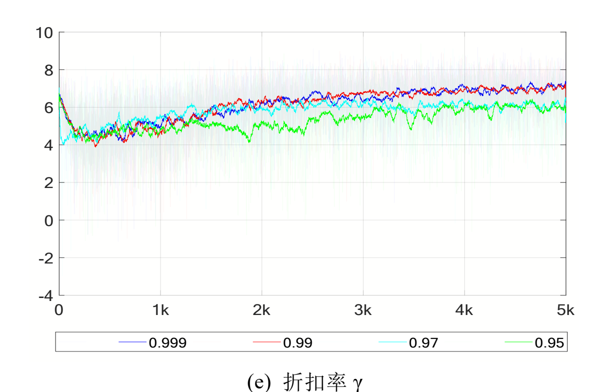
(b) 学习率



(c) 最小训练批量 mini_batch



(d) 目标网络更新周期 N



(e) 折扣率 γ

图 1　各超参数的验证结果

Fig. 1　Verification results of each hyperparameter

3.2　DRL 模型求解问题集

本节将本文提出的 DRL 模型运用至两种问题集的 10 个算例上，实验分为模型训练阶段和测试阶段。在训练阶段，将 DRL 模型分别在不同工期分布下的 10 个算例分别进行 5000 次仿真训练，模型的超参数取值策略如表 4 所示，并将各算例训练完成的模型保存本地。在模型的测试阶段，将 10 个算例所训练完成的 10 个模型分别在对应算例的 5 种工期分布下进行 50 次仿真调度求得平均的 TTC。

表 4　超参数组合

Tab. 4　Hyperparameter combinations

超参数	取值
T_max	5000
ε_{min}	10^{-5}
ε_{rate}	0.9999
rate	10^{-3}
M	10^{5}
N	100
mini_batch	128
γ	0.99
L	20

图 2 为算例 MP30_5_5 在五种工期分布下的训练过程，其中横坐标为多项目的总拖期成本，纵坐标为模型的训练回合。可以看出对于方差相对较小的 U1、B1 的 TTC 迭代曲线处于最下方且波动最小；对于方差中等的 U2、B2 的 TTC 迭代曲线处于中间位置且波动中等；对于方差较大的 Exp 的 TTC 迭代曲线处于最上方位置且波动较大，表明总拖期成本的期望水平随着活动工期不确定程度的增加而增大。

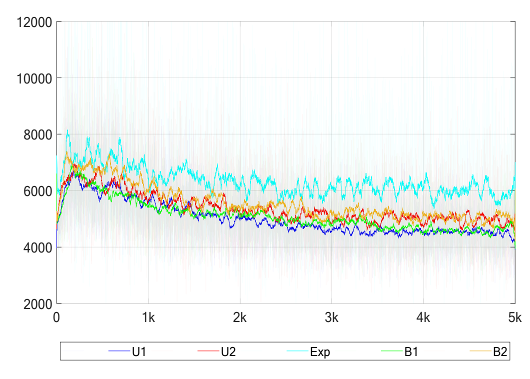


图 2　不同工期分布下的 TTC 迭代过程

Fig. 2　TTC iteration process under different duration distributions

对于算例 MP30_5_5 训练完成的模型的测试阶段，将训练完成模型与动作空间中 15 种单一调度规则在对应工期分布下分别进行 50 次仿真调度求得平均的 TTC 如图 3 所示。由图 3 可以看出 DRL 算法克服单一规则的短视性，在动态环境中获得更好的调度结果。进一步的选取 15 种规则在 5 种

工期分布下表现最好的规则与 DRL 模型所得的结果进行对比并以改进率 improve 作为评价指标，如式(11)所示。强化学习改进率如表 5 所示。

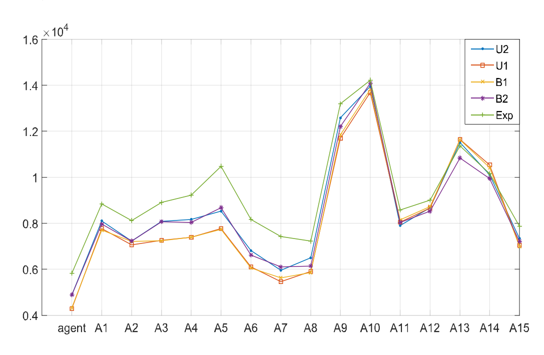


图 3　不同工期分布下各动作的 TTC

Fig. 3　TTC of each action under different duration distributions

表 6 为优先规则算法和 DRL 模型调度在完成 50 次多项目调度过程的运行时间。可以看出与优先规则调度算法相比，训练后的模型可以根据环境的当前状态快速作出最优调度决策，速度相当于优先规则算法。

MP30_5 与 MP90_2 问题集在不同工期分布下 50 次调度的平均 TTC 值如表 7 所示。对比文献[9]，其中 MP30_5 的在 U1 分布下的结果本文较差，而 MP90_2 的结果本文较优。

$$improve=\frac{Best-agent}{Best}\times 100\% \tag{11}$$

表 5　强化学习改进率

Tab. 5　Improvement rate of reinforcement learning

分布	agent	Best rule	Improve(%)
U1	4286.25	5455.55	21.43
U2	4890.75	5952.62	17.84
Exp	5825.32	7219.41	19.31
B1	4319.49	5632.16	23.31
B2	4889.91	6102.43	19.87

表 6　单一规则和 DRL 算法调度时间

Tab. 6　Single rule and DRL algorithm scheduling time

Scheduling time	MP30_2	MP30_5	MP30_10
Single rule	2.24	2.64	3.16
DRL	2.58	3.04	3.98

表 7　问题集不同分布下 DRL 结果

Tab. 7　DRL results under different distributions of problem sets

TTC	1	2	3	4	5
MP30_5	1467.57	1748.06	1422.57	1783.27	2279.03
Mp90_2	1001.58	1524.73	1025.81	1439.44	2497.44

3.3　不确定到达时间

在实际情况中项目的到达时间往往会因局部资源的缺乏而与预计的到达时间产生偏差。因此本节研究了 DRL 模型在项目到达时间不确定环境下对多项目总拖期成本的影响，其中项目活动的工期为常工期分布，项目到达时间服从以下分布：

$$ST_i=\begin{cases}U1 & a=0\\ U2 & a=1\\ Exp & a=2\\ B1 & a=3\\ B2 & a=4\end{cases} \tag{12}$$

式(12)中的分布类型特征与 3.2 工期分布相同；a 为[0,4]的随 $TC_i=(FT_i(J_i+1)-ST_i-cpl_i)\times c_i$ 机整数，例如当 a=0 时表示项目 i 的到达时间为服从 U1 分布的随机数。

多项目到达时间不确定所产生的状态组合小于不确定工期情况，因此 DRL 模型的训练回合设置为 5000 次且模型的超参数组合与 3.2 节相同。图 4 为模型训练过程的总拖期成本 TTC 的变化曲线，在训练的过程中 TTC 随着训练的回合的增加不断减少并趋于稳定，表明模型进行了有效的训练。

同时现有文献并未有对不确定项目到达时间的 DRCMPSP 的研究，因此将训练完成的模型进行 100 次仿真实验后所得的平均 TTC 值 3022.01，与 15 种规则中最优规则的 TTC 值 4973.75 进行对比，其改进率为 64.6%。

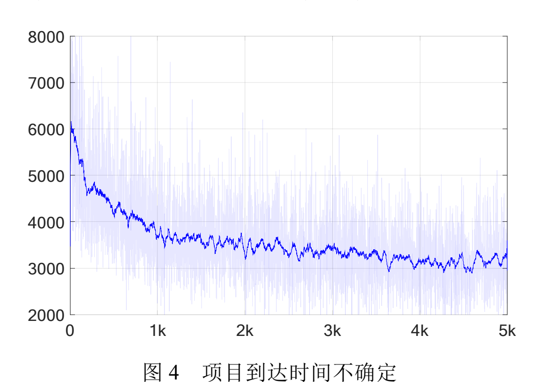


图 4　项目到达时间不确定

Fig. 4　Uncertain arrival time of the project

4　结束语

本文首次将深度强化学习运用至多项目调度问题，在此基础上提出基于 DRL 的分布式多项目动态调度模型，以实现随机工期下分布式多项目调度问题总拖期成本最小化的目标。并搭建了智能体交互的仿真环境，以算例 MP30_5 和 MP90_2 问题集中的算例进行仿真实验，一方面对模型的超参数取值策略进行灵敏度分析，另一方面对通过算例对模型进行训练和测试。结果表明本文所提出的 DRL 模型对于实现分布式多项目在随机环境下的动态调度有一定优势，训练好的模型在决策阶段的作出策略的速度与优先规则相差无几，同时能够有效降低随机分布式多项目调度所需的总拖期成本，拓展了深度强化学习在随机性项目调度问题上的运用。

参考文献：

[1] Lova A, Tormos P. Analysis of Scheduling Schemes and Heuristic Rules Performance in Resource-Constrained Multiproject Scheduling [J]. Annals of Operations Research, 2001, 102 (1-4): 263-286.

[2] Lee Y H, Kumara S, Chatterjee K. Multi-agent based dynamic resource scheduling for distributed multiple projects using a market mechanism [J]. Journal of Intelligent Manufacturing, 2003, 14 (5): 471-484.

[3] Davari M, Demeulemeester E. Important classes of reactions for the proactive and reactive resource-constrained project scheduling problem [J]. Annals of Operations Research, 2019, 274 (1-2): 187-210.

[4] Satic U, Jacko P, Kirkbride C. Performance evaluation of scheduling policies for the dynamic and stochastic resource-constrained multi-project scheduling problem [J]. International Journal of Production Research, 2020, 60 (4): 1411-1423.

[5] Wang Yanting, He Zhengwen, Kerkhove L P, *et al.* On the performance of priority rules for the stochastic resource constrained multi-project scheduling problem [J]. Computers & Industrial Engineering, 2017, 114 (DEC.): 223-234.

[6] Chen Haojie, Ding Guofu, Zhang Jian, *et al.* Research on priority rules for the stochastic resource constrained multi-project scheduling problem with new project arrival [J]. Computers & Industrial Engineering, 2019, 137 (2): 106060-.

[7] Song Wen, Xi Hui, Kang Donghun, *et al.* An Agent-based Simulation System for Multi-Project Scheduling under Uncertainty [J]. Simulation Modelling Practice and Theory, 2018, 86 (11): 187-203

[8] Tosselli L, Bogado V, E Martínez. A repeated-negotiation game approach

to distributed (re) scheduling of multiple projects using decoupled learning [J]. Simulation Modelling Practice and Theory, 2019, 98 (4): 101980.

[9] 刘东宁, 徐哲. 基于多优先规则启发式的分布式多项目随机调度 [J]. 系统工程理论与实践, 2021, 41 (12): 3294-3303. (Liu Dongning, Xu Zhe. A stochastic scheduling for distributed multi-project with multi-PR heuristic [J/OL]. Systems Engineering-theory & Practice, 41 (12): 3294-3303.)

[10] 韩忻辰, 俞胜平, 袁志明, 等. 基于 Q-learning 的高速铁路列车动态调度方法 [J]. 控制理论与应用, 2021, 38 (10): 1511-1521. (Han Xincheng, Yu Shengping, Yuan Zhiming, *et al.* High-speed railway dynamic scheduling based on Q-learning method [J]. Control Theory & Applications, 2021, 38 (10): 1511-1521.)

[11] Waschneck B, Reichstaller A, Belzner L, *et al.* Deep reinforcement learning for semiconductor production scheduling [C]// SEMI Advanced Semiconductor Manufacturing Conference (ASMC) . 2018.

[12] Lin Chuncheng, Deng Derjiunn, Chih yenling, *et al.* Smart Manufacturing Scheduling with Edge Computing Using Multi-class Deep Q Network [J]. IEEE Trans on Industrial Informatics, 2019, 15 (7): 4276-4284.

[13] Liu Chienliang, Chang Chuanchin, Tseng C J. Actor-Critic Deep Reinforcement Learning for Solving Job Shop Scheduling Problems [J]. IEEE Access, 2020, PP (99): 1-1.